Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

ELSEVIER Electronic Notes in Theoretical Computer Science 93 (2004) 183–201
www.elsevier.com/locate/entcs

Mathematical, Interactive Exercise Generation from Static Documents

Manolis Mavrikis[1]

School of Mathematics,
The University of Edinburgh,
Scotland - UK

Alberto González Palomo[2]

Toledo,
Spain

Abstract

We present an exercise description language suitable for the representation of interactive exercises which involve mathematical entities, and have complex structures. One of the goals of the language is to provide primitives for support of interactive environments where adaptive presentation of steps and feedback is required. Using this scheme, exercises can be generated from static documents, in what can be seen as a layered scheme: the interactivity layer is applied over the static layer, yielding an interactive version of the content. This significantly speeds up the authoring process, makes the document management process easier, and helps in reusing the document. We believe that the information model presented in this paper, as well as the proposed way of deployment, may provide grounds for new extensions of today's standards helping to overcome some of the limitations that they currently have.

Keywords: mathematical interactive exercises, description language, QTI and OMDOc specification.

[1] Correspondence: School of Mathematics, JCMB - Kings Buildings, UoE, EH93JZ, Edinburgh, Scotland. Email: M.Mavrikis@ed.ac.uk

[2] Email: Alberto.Gonzalez@matracas.org. The author is currently employed part-time as research assistant (student) at the University of Saarland/DFKI, Germany, but this work was done on his own without their supervision or support.

doi:10.1016/j.entcs.2003.12.025

1 Introduction

Authoring content for any system is time consuming and often conducted in a proprietary format that makes reusability hard if not impossible. Appropriate specifications, such as the IMS QTI [17], are being developed to represent questions, tests, and activities and enable their exchange between educators, and interoperability among different learning management systems. However, QTI was not explicitly developed with mathematics in mind (see [3,10,16]) and ignores many significant problems that authors of mathematical activities face; the fact that they are often complex with multiple parts, the need to use multiple notations and allow entry of mathematical expressions, the fact that answers often entail mathematical entities beyond simple numbers or strings yielding the need to handle accuracy and precision of numbers.

Fortunately, representations such as OpenMath[5], MathML[6] and OMDoc [11] help in publishing mathematics and provide support for delivering, re-using and exchanging mathematical expressions. OMDoc, in particular, covers some of the limitations mentioned before since, apart from the representation of the mathematical entities, it provides structures for the meaning of the document itself, thus offering machine understandable mathematical documents. This is something very important for systems which are becoming more adaptive and intelligent.

On the other hand, exercises in OMDoc have no particular structure and were initially designed to address multiple choice questions and hints in the form of remarks disallowing more complex structures. The authors of [8] describe a generic exercise scheme that takes into account the multiple purposes that an exercise can serve (for instance to motivate theory or the possibility of using variations of an exercise to present various aspects of theory). These are important issues to take into account for the metadata of any representation as they can provide extra support for adapting the material to student needs. In addition, [8] describes some of the problems that we also faced trying to author exercises for first year science and engineering students. An exercise can often have more complicated structure than the ones anticipated by OMDoc, containing multiple parts, and expecting students to interact with it and receive feedback while they try to complete it. In order for the feedback to be effective it needs to be adaptive according to student errors and misconceptions. Although QTI provides some of these functionalities it is considered important [13,4] to provide exercises with which also adapt the presented parts and feedback according to students' errors in (often) different parts of the question. Using external systems to support this process requires semantically tagged mathematics that QTI does not support.

Of course, there are tools and systems that in their own language format

address some of these issues individually. For example, CUE[7] has its own format which allows mathematical questions with multiple steps as well as editing tools to facilitate authoring. The open-source system AIM [1], has its own language using a mixture of AIM-specific flags and Maple-like code to achieve randomisation of questions and adequate answer evaluation. Similarly, the commercial MapleTA[12] offers a full-fledged web-based system for creating questions and assessing student responses using Maple on the background. In WaLLiS[14] a similar approach was taken. Java Servlets and Beans were used to create questions from random values and AIM's evaluation procedures were integrated to evaluate student input. The philosophy of WaLLiS is closer to tutoring than assessment systems hence interaction, adaptive presentation and feedback delivery were considered very important. On the other hand, in the past, the interactive parts had to be authored one-by-one (only aided by templates) concluding to hours of development and activities that were really hard to maintain and reuse. Other systems, like the QTI-compliant version of SToMp [9], following QTI means loosing some of the functionality that their developers would like to have such as proper testing of precision and accuracy of numeric answers, and the ability to randomise values in questions. These are only a few examples of systems which are used in everyday teaching where an author is 'locked' in the format that these systems provide. The need of a common structure for the representation of questions that can support all of them is apparent. This will ease authoring (as common tools will be developed), document management but more importantly the exchange of content between authors and learning management systems.

This paper, presents an information model which achieves in explicitly representing the mathematical entities in questions, as well as allowing the authoring of more complex structures and interactive exercises with multiple parts and adaptive feedback. The information model has occurred from the need to author such questions for the domain of algebra and calculus for science and engineering teaching but we believe it encompasses many similarities with exercises from other domains and (probably with some extensions) can be used to represent activities that require the learner to work with even more advanced concepts (such as proofs).

The main focus of this paper is the information model and the examples which show how can one bypass the limitations mentioned before. We would like to make clear here that, although we present examples from the working system WaLLiS this is only to facilitate our discussion. We will neither present here details of WaLLiS' architecture (some of these can be seen in [14]) nor cover details of the whole information model that would require an enormous

amount of unnecessary explanation[3]. Doing so, would be like proposing a new standard, which is not our intention. Our aim is to motivate discussion with other interested parties to reach a common structure that would, at least, permit transformations from one format to the other without loss of functionality.

2 The exercise description language.

The information model presented here has been partly inspired from QTI and the exercise model in [8]. The latter is currently being implemented in ActiveMath [15], following it strictly by nesting elements to achieve complex structures. Here, we have paid more attention on how we can ease the authoring process by providing an 'interactive layer' over the static layer for the mathematical entities (section 3.1). We have also added support for randomised questions (section 3.2) and, in contrast to the explicit hierarchical structure of that model, our language consists of top-level nodes that are linked by identifier references (section 3.3). We have found this approach to be more flexible and understandable especially when writing large exercises. Apart from helping us writing exercises by hand at this stage, where authoring tools are not yet available, it also helps in managing and storing (for example, in a database) complex structures which need to be maintained over long periods of use. If items or part of questions are difficult to locate, read and modify, they are less likely to be kept up to date. Our shallow tree structure helps in that.

```
exercise := (variables?,item+)
item := (variables?, material*, responses?, answer?, map_action*)
material := (text|image|applet|other)+
responses := (response?)
response := (blank|choice|hotspot)
map_action := (cond?,linked_item*,system_messages?)
```

Fig. 1. Excerpt of the schema for the presented structure. Notice that `items` are not nested but linked through the `map_action` element (see section 3.3)

It is worth mentioning again that the information model presented, the naming conventions used, and the examples are from the working system WaLLiS but we only use them here as a way of explaining this structure and how the ideas can be used in other specifications.

[3] further details will be made available online at http://www.maths.ed.ac.uk/wallis/format

In this model (see figure 1), an `exercise` is composed of several `<item>` elements which represent the different parts of the exercise as well as the steps (e.g. receiving feedback, showing a part of the exercise etc.) in the system-student interaction. The important elements that an `<item>` contains are the following:

- `<material>`: This (similarly to QTI `<material>` or OMDOC's `<CMP>` element) is the container for any supported content (e.g. text, applet) that is to be displayed to the user, such as the problem statement or the feedback .
- `<response>`: Contains information about the type of response expected for this item. Multiplicity represents exercises where the user is expected to provide more than a simple answer (e.g. a matrix, a fraction etc). `<response>` elements have a `for` attribute which can be used to match identified parts of the text and to be replaced by interactive elements (blanks, combobox etc.) (see example in section 3.1)
- `<variables>`: Specifies the variables that are going to be used throughout the item either to randomise the question or ease the authoring process (see section 3.2).
- `<map_action>`: Specifies what action to take when user-interface events occur, such as the student answering, or asking for a hint or the solution. By referring to other items the system can deliver multi-part questions, and adaptive feedback. The action is executed only if the range of tests that the `<cond>` element includes are satisfied (see example in section 3.3).
- `<answer>`: Contains information about the correct answer. Similarly to the `<response>` element it has a `for` attribute to match each response. This is not to be confused with the *solution* of an exercise as the latter can be a whole new item as declared in the `map_action type="onSolution"` element (see section 3.3).

The following sections explain these and other important parts of the model in more detail through examples that show how it can represent questions that contain mathematics, how variables are used to randomise them and how we achieve feedback adaptation and communication with external systems.

3 Examples

3.1 Representing mathematics

Let us first focus on some problems considering the semantics of exercises and the mathematical entities that they contain. Although one could think ways

around developing questions that expect the user to input, for instance, a matrix or an integral using QTI's format, the semantics of the question are notably reduced making it impossible for an engine to know what exactly these entities are. This is a significant problem for intelligent, adaptive systems that need to communicate with external systems (such as a Computer Algebra System or a proof planner) and provide the ability to adapt their content, based on students' profile, and deliver feedback based on their mistakes.

Therefore, the text elements can contain OpenMath for the mathematical expressions. This eliminates the problem of the semantical representation of the entities involved and allows the system to deal with its representation separately. This approach works equally well for MathML but, for now, we are using OpenMath for writing the expressions, to avoid getting tied into a specific notation. This way, the exercises are easier to translate to other languages. For instance, in an implementation of this model for WaLLiS, in order to produce an interactive exercise to be displayed in a browser window, transformations are applied to the XML source file to produce XHTML (with MathML) for Mozilla browsers or HTML for other browsers.

For an example, imagine an exercise that expects learners to input a matrix (see figure 2). The steps are:

(i) Annotate an arbitrary matrix like $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$, marking the replaceable elements with ‘‘id’’ attributes.

```
<OMOBJ xmlns="http://www.openmath.org/OpenMath">
  <OMA>
    <OMS cd="relation1" name="eq"/>
    <OMV name="A"/>
    <OMA>
      <OMS cd="linalg2" name="matrix"/>
      <OMA>
        <OMS cd="linalg2" name="matrixrow"/>
        <OMI id="m_1_1">1</OMI>
        <OMI id="m_1_2">2</OMI>
      </OMA>
      <OMA>
        <OMS cd="linalg2" name="matrixrow"/>
        <OMI id="m_2_1">3</OMI>
        <OMI id="m_2_2">4</OMI>
      </OMA>
    </OMA>
```

```
  </OMA>
</OMOBJ>
```

Of course, one could omit the values of the `<OMI>` elements as they play no role and they are going to be replaced by input fields but it helps in copying-pasting questions from other systems or automatically exporting them from Computer Algebra Systems (CAS) that support OpenMath. In addition there is some scope to include the actual correct answer here in the elements to be matched but, so far, this happens with the `<answer>` element for consistency with other formats that would not follow our layered approach.

(ii) The next step is to use the ‘‘`for`’’ attribute of the `<response>` element in order to map each replaceable element:

```
<responses>
   <response type="num" id="fib-1">
      <blank for="m-1-1" maxchars="3"/>
   </response>
   <response type="num"  id="fib-2">
      <blank for="m-1-2" maxchars="3"/>
   </response>
   <response type="num" id="fib-3">
      <blank for="m-2-1" maxchars="3"/>
   </response>
   <response type="num" id="fib-4">
      <blank for="m-2-2" maxchars="3"/>
   </response>
</responses>
```

In this case, we replace parts of the expression with blank input fields, thus producing a multiple, “fill in blanks” exercise where order is important.

(iii) The resulting MathML is the following, rendered as $A = \begin{pmatrix} \square & \square \\ \square & \square \end{pmatrix}$:

```
<m:math display="block">
  <m:mi>A</m:mi><m:mo>=</m:mo>
  <m:mfenced><m:mtable>
      <m:mtr>
        <input type="text" id="fib-1" size="3" value=""/>
        <input type="text" id="fib-2" size="3" value=""/>
      </m:mtr>
```

```
        <m:mtr>
          <input type="text" id="fib-3" size="3" value=""/>
          <input type="text" id="fib-4" size="3" value=""/>
        </m:mtr>
      </m:mtable></m:mfenced>
  </m:math>
```

By separating the content from the presentation we have managed to ease authoring. Using for instance a LaTeX to OpenMath or MathML converter one can use previously written material and then by hand match the blanks. In the future this format could help tools such as QMath (http://www.matracas.org) , TexMacs (http://www.texmacs.org) or a GUI to make this process even more transparent and easier for less experienced authors.

A similar approach could be used to match blanks for a text passage and replace them with comboboxes or blanks using a structure such as:

```
<text>Fill this <with id="l1"/> and <with id="l2"/></text>
```

As this is still work in progress and happens less often in mathematics, we will not develop further this point and the reader is referred to the online examples.

3.2 Employing Variables For Questions

We have found that, in certain domains and types of questions, instead of rewriting several versions of the same question it is often easier to use variables that are manually changed (thus producing a new exercise) or even better randomised by the system. In addition, variables could be used to ease other authoring aspects such as feedback or solution adaptation based on students' input. For instance, AIM and MapleTA allow for that kind of randomisation by employing Maple on the background. Of course, this bonds the activities of these systems with their specific language and unfortunately there is no built-in support for something like that in QTI (neither OMDoc) and reusing even the statements of such exercises would be impossible.

Therefore, we introduce an element, called `<variables>`, that can be used to define activity variables, how their value is to be generated, and how it is going to be formatted. For instance, the following variables

```
<var id="m" minvalue="1" maxvalue="3"/>
<var id="f_tmp" minvalue="0" maxvalue="1"/>
<var id="f">2*f_tmp-1</var>
<var id="n" minvalue="1" maxvalue="3"/>
```

```
<var id="z" minvalue="1" maxvalue="10"/>
<var id="a">m+4*n*f</var>
<var id="b">4*(m-n*f)</var>
<var id="c">4*m+n*f</var>
<var id="d">5*z</var>
```

are used to create random coefficients for the conic of figure 2.

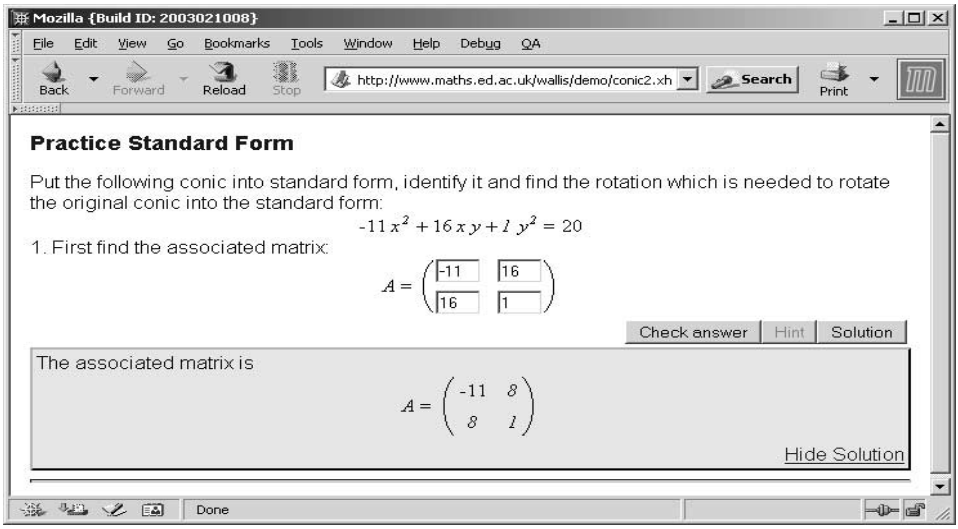

Fig. 2. A part of an exercise that expects students to input a matrix; the solution is also displayed after the user's request.

Note that due to the lack of a standard for such expressions, at the moment, we still use common programming expression syntax (e.g. `2*a^2`) (although in anticipation of a more widespread use, MathML or better OpenMath should be used to define these).

Upon declaration, variables in an OpenMath would be overwritten with the random generated value. Thus the OpenMath representation of the expression in figure 2 is

```
<OMOBJ xmlns='http://www.openmath.org/OpenMath'>
<OMA>
  <OMS cd="relation1" name="eq"/>
  <OMA>
      <OMS cd="arith1" name="plus"/>
      <OMA>
        <OMS cd="arith1" name="times"/>
        <OMV name="a"/>
```

```
        <OMA>
          <OMS cd="arith1" name="power"/>
          <OMV name="x"/>
          <OMI>2</OMI>
        </OMA>
      </OMA>
     <OMA>
       <OMS cd="arith1" name="times"/>
       <OMV name="b"/>
       <OMV name="x"/>
       <OMV name="y"/>
     </OMA>
      <OMA>
        <OMS cd="arith1" name="times"/>
        <OMV name="c"/>
        <OMA>
         <OMS cd="arith1" name="power"/>
         <OMV name="y"/>
         <OMI>2</OMI>
        </OMA>
    </OMA>
  </OMA>
  <OMV name="d"/>
</OMA>
</OMOBJ>
```

The values of these variables can also be used (using the `<var_value>` element) inside the `<text>` element to adapt (for instance) the feedback. Similarly, variables can be used for the conditional tests for mapping the action of the system to student's response (see the example in 3.3.1).

This way, the use of the background system is irrelevant and different systems can use different tools to generate random numbers or evaluate the expressions. The use of variables is still work in progress but recent similar efforts (eg. [2]) are looking into having a much wider adoption of variables inside the QTI framework. We hope that other formats would be able to follow. Of course, there are important aspects to take into account when using variables. From an author's perspective, one important issue is that certain random values can lead to inconsistent questions and appropriate tools are necessary to simplify or check for validity of the random numbers. Another issue is the scope of a variable which we currently address by usual programming conventions of local (the ones defined under `item` elements) and global

variables (the ones defined under the root of the `exercise` element) . These are rather subjective decisions which should be thought carefully with other interested parties. On the other hand, having an appropriate way to represent them is a necessary precursor for further discussions in order to achieve a general format that could help in exchanging questions.

3.3 Enabling interaction and complex structures

As we mentioned before, it is important to be able to achieve complex structures and adequate and adaptive interaction. Exercises can be used in many ways, from self-learning to knowledge assessment, but whether one decides to nest elements (as in [8]) or use the information model as described in section 2 there is a need to provide rules for adapting the items that are to be displayed according to the learner's actions, responses and state of mind. This includes both the presentation of steps (or parts) of the question as well as feedback, hints and answers that the learner will receive.

To achieve this adaptation, we use the `<map_action>` element which takes a `type` attribute that specifies what the action refers to (e.g. onAnswer, onHint, onSolution) etc. In addition, it contains a `<cond>` element that represents the tests that should be satisfied for the action to take place. We will not elaborate here on the various tests that we use (some of these are inspired by the same tests that QTI uses). For example, the elements `<num_equal>`, `<num_lt>`, `<num_gt>` are used to test for equality and inequalities. In addition there are `<not>`, `<or>` elements to create Boolean 'AND', 'OR' operations (examples of how these are used for QTI can be seen in [17]). The difference here is that instead of just referring only to feedback items that can only contain material the `<map_action>` element includes a `<linked_item>` element that links to another item thus allowing any complex structures to exist. Similarly to [8] the `<system_messages>` element provides information (such as marking) to the system. Some examples will help to clarify these better.

3.3.1 Adapted feedback

The following excerpt defines two misconception that learners usually have that could be used for adapting the hint that they will receive.

```
<!-- students often think that the A_1_2 element of the associated
     matrix of a quadratic ax^2+ b*x*y + c*y^2 is b instead of b/2 -->
<map_action type="onHint">
 <cond>
  <num_equal response_id="fib-2">
     <var_value>b</var_value>
```

```
   </num_equal>
  </cond>
  <linked_item xref="misconception1"/>
</map_action>
```

When learners demonstrate, for example the first misconception, they would receive the following feedback

```
<item id="misconception1"><material>
     <text>Have a look at the formula again. The value of
     A12 should be half the value of the coefficient of x*y</text>
</material></item>
```

3.3.2 Multiple parts and more complicated structures

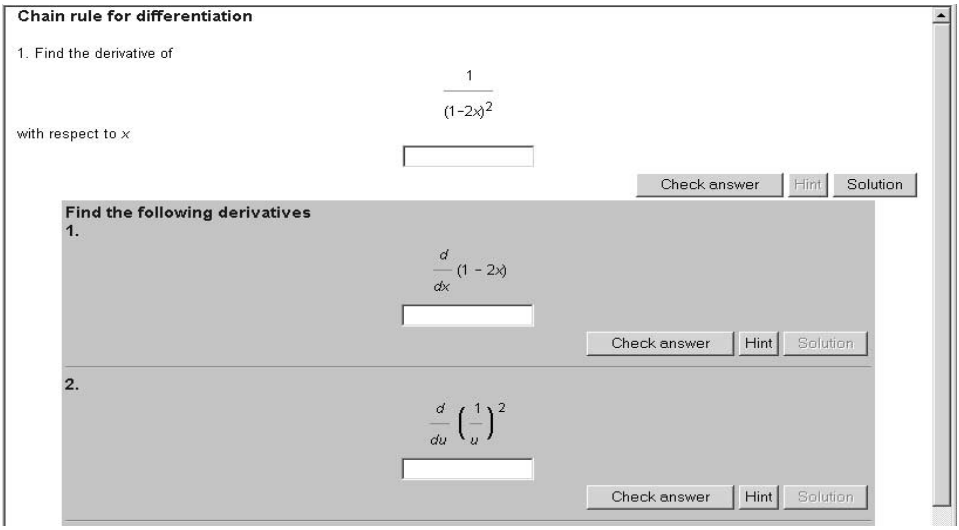

Fig. 3. A differentiation problem with two sub-items as a hint.

The (hint) item in the previous section is on its own already very effective as it provides the ability to provide feedback on the exact misconception that students have. On the other hand there are certain cases where instead of a hint we would like to provide a whole new exercise, a subproblem for the student to solve, that would help them understand their problem better. For example, figure 3, shows a question where students are supposed to find the derivative of an expression. When they ask for a hint they get another item that breaks the problem into two sub-parts.

Due to space limitations and in order to show the difference with [8] instead of the XML that represents the problem in figure 3 we will provide here

the source of the example represented in a nested multiple choice question in Tables 2,3 of [8] which addresses the following question:

Given a set of natural numbers with an addition operation and zero, which of the following structures is it?

◯ Group

◯ Monoid

◯ None of these

Its XML source would be

```
<item id="MCQ">
  <material><text>
     Given a set of natural numbers with an addition operation
     and zero, which of the following structures is it?
  </text></material>
  ...
  <responses>
    <response id="quest1">
      <choices>
        <choice>
          <material><text> Monoid </text></material>
        </choice>
        <choice>
          <material><text> Group </text></material>
        </choice>
        <choice>
          <material><text> None of these </text></material>
        </choice>
      </choices>
  </response>
 </responses>
  ...
</item>
```

The item contains a response element that represents the MCQ. The details of the MCQ representation are not so important as it is trivial to transfer it to any other format but, although it seems verbose, it helps in being consistent

with the rest of the model and achieving having mathematical entities even inside the MCQs. What is important to notice here is the action elements that would be included in this item. One of these:

```
...
 <map_action type="onAnswer">
   <cond>
     <equal response-id="quest1"> B </equal>
   </cond>
   <linked_item xref="one-more-chance"/>
</map_action>
 ...
```

links to another item with id `one-more-chance`. Replying `B` would give raise to another subproblem

```
<item id="one-more-chance">
    <material><text> No, it's not a group.
                     But if you find why it's wrong you'll get
                     half the marks
              </text></material>

    <responses>
        <response id="ch1">
          <choices>
            <choice>
                 <material><text> It has no units </text>
                 </material>
            </choice>
            <choice>
                 <material><text> It has no inverses </text>
                 </material>
            </choice>
          </choices>
        </response>
   </responses>
    ...
</item>
```

which appears in figure 4. This subproblem could in turn have other subproblems and so on, achieving any nested structure.

No, it's not a group. But if you find why it's wrong you'll get half the marks.

◯ It has no unit

◯ It has no inverses

Fig. 4. An item that appears when the previous one is wrong.

4 Transforming the document

We will not elaborate here on transformation details from the XML source files to final output rather than mention that these are possible using techniques such as XSLT. For instance, for WaLLiS, the transformation results appear on a browser page that includes a form for the interaction (see fig. 2). Our format, by separating content from presentation, is easily transformed to any desired output. For example in the future, we will investigate how the information model of XForms [18], the emerging standard for web forms, could be employed, since it has lots of similarities and also separates content from presentation.

In addition the format lends itself to being completely separate from implementation and architectural issues. For instance, fo the self-practice examples of WALLIS (where security is not important) the final HTML (or XHTML) page includes hidden elements (eg. the solution or the hints) and JavaScript that takes care of the interaction. For example, in figure 2, the user just exhausted all the hints that they could get, thus being permitted to get the solution which was hidden below the rest of the exercise. This way the amount of traffic (submitting and retrieving pages) is reduced. Other systems could use server side interpreters to take care of the actions, especially when this entails questions with which the learner is formally assessed.

5 Linking to external systems

The interactive activities that WaLLiS provides, communicate with a CAS to interpret steps of the user interaction or input. This is achieved transparently, without the user starting an external system in a different window. The information model presented here and the variable extensions were proven more than adequate and really useful for this purpose. Using a `<cas_test>` element inside the `<cond>` elements content authors can employ more advanced tests on the expressions that a user enters. For example:

```
 <map_action type="onAnswer">
   <cond>
     <cas_test for="ans0">‘aim/Diff/WallisTest‘(Q);</cas_test>
   </cond>
   <linked_item xref="DIFF_CHAIN1_CORRECT"/>
  <system_msg>
     <completed/>
  </system_msg>
</map_action>
```

By integrating AIM's tests, answers can be evaluated in depth. The tests can also assert misconceptions on common tasks (e.g. differentiation, integration) or distinguish between the written forms of mathematical expressions which are algebraically equivalent (e.g simplified, factored, unfactored). Of course, these specific tests defeat the purpose of having a structure which could help in exchanging questions, since different systems may be employing different CAS. Nevertheless, even if the tests are specific, a common format such as the one presented here could help in, at least, exchanging the overall exercise format. Further research could look into reaching a system-independent markup for these tests that other interested parties could use.

6 Links to OMDoc and QTI

As we mentioned before, presenting all the details of the information model here is considered beyond the scope of this paper especially at this preliminary phase. We are still working on finalising some of the aspects that fit the activites in WaLLiS and other systems. We provide the examples and the discussion as inspiration for further collaboration with interested parties that could lead in a shared format. For instance, the exercise description module of OMDoc could benefit from the ideas proposed here, namely the way we match OpenMath expressions, how we link items instead of nesting them, the variable support and the conditional executions that can provide adaptive exercises.

Similarly, the 2nd version of QTI is currently under development, and we strongly believe that the issues raised here are important and can be easily taken into account in order to be able to provide exercises that are transformable, if not exchangeable, between systems. First of all, there is an apparent need for supporting mathematical notation as we do here and the use of variables for randomising questions as in [2]. Without this support, the questions that contain mathematics are susceptible to an enormous semantic reduction that would make further authoring and maintenance impossible.

In addition, QTI's *displayfeedback* element is similar to our *map_action type="onHint"* element but refers only to feedback elements restricting complex structures. In our view, feedback and solution elements are unnecessary since there is often a need for them to be completely new items as shown in the previous examples. Finally, QTI's *displayfeedback* element cannot take any preconditions thus making feedback adaptation impossible. It is obvious that any transformation from the format presented here to QTI would certainly lose functionality.

7 Reuse of these techniques in other contexts

The needs of interactive exercises as presented here have much in common with those of other systems which involve the manipulation and evaluation of mathematical input. With the use of such external systems we can check for all sorts of errors that students manifest and provide adaptive feedback. We briefly showed an example with a CAS but similar communication could be achieved with other intelligent systems or automated provers.

In particular, there is a need to select specific parts of an expression (the proof) to operate with them, and then to check automatically the validity of each step. In an e-learning system (limited to the domain of the automated prover) the point is not only to grade the student's answer but to provide guidance when the computer can not derive a step from the previous one. The system would take note of the failure, and if in teaching mode (as opposed to grading mode) it would display information intended to help the student complete the proof.

8 Conclusion and future work

We described an information model that achieves a clear separation between the mathematical entities, and the interactive and presentation layer of the generated questions, thus overcoming the inefficiencies that other formats have. The use of variables makes authoring even easier and the fact that the interactive exercises are generated from static documents helps interoperability, reusability and ease of maintenance. The shallow tree structure helps in managing and databasing the document as well as reusing the feedback elements or independent parts of the exercise in different exercises. Simultaneously, the format allows authoring any complex exercise structure, and provides the ability to deliver feedback and steps of the exercise in an adaptive manner. These could not be represented with any other information model so far. Of course, the format presented here needs further refinement

and we do not describe it as a proposed standard but rather as a structure and inspiration that other formats could encompass, leading to a common format where sharing and transforming from one format to the other would be possible without loss of functionality.

9 Acknowledgments

We would like to thank Dr Antony Maciocia for his help in this paper and generally in deriving the information model of WaLLiS, Prof. Cliff Beevers and Dr. Helen Ashton for the information on CUE and multi-part questions, Dr. Dick Backon for the information on the 'random numbers' extensions for QTI and the discussion on the problems faced when adopting QTI for SToMP's content, and the QTI-list members for their comments and discussions.

References

[1] AIM, http://aimmath.sourceforge.net.

[2] Bacon, D., *Suggestion for QTI support for question variables, expressions and graphs*, QTI working papers in: http://ford.ces.strath.ac.uk/QTI/.

[3] Bacon, D., *IMS question and test interoperability*, LTSN CAA series. See http://ltsn.mathstore.ac.uk/articles/maths-caa-series/ (2003).

[4] Beevers, C., M. Youngson, G. McGuire, D. Wild and D. Fiddes, *Issues of partial credit in mathematical assessment by computer*, Alt-J **7** (1999).

[5] Caprotti, O. and A. Cohen, *Draft of the open math standard, OpenMath consortium,* http://www.nag.co.uk/projects/openmath/omstd/ (1998).

[6] Carlisle, D., P. Ion, R. Miner and N. Poppelier, *Mathematical markup language, version 2.0,* http://www.w3.org/tr/mathml2/ (2001).

[7] CUE, http://www.calm.hw.ac.uk/cue.html.

[8] Goguadze, G., E. Melis, C. Ullrich and P. Cairns, *Problems and solutions for markup for mathematical examples and exercises*, Mathematical Knowledge Management: Second International Conference, MKM 2003, Bertinoro, Italy, 2003 (2003).

[9] Hunt, J. and B. R.A., *SToMP: A Dynamic Approach to Courseware Development*, EDMEDIA 97 (1997).

[10] IMS-QTI-LIST, http://www.topica.com/lists/IMSQTI.

[11] Kohlhase, M., *OMDoc: Towards an OpenMath Representation of Mathematical Documents,* FR Informatik, Universitaat des Saarlandes (2000).

[12] MapleTA, http://www.maplesoft.com/mapleta.

[13] Mavrikis, M. and A. Maciocia, *Incorporating assessment in a web-based ILE*, LTSN CAA series. See http://ltsn.mathstore.ac.uk/articles/maths-caa-series/.

[14] Mavrikis, M. and A. Maciocia, *WaLLiS: a web-based ILE for science and engineering students studying mathematics*, Workshop of Advanced Technologies for Mathematics Education in 11th International Conference on Artificial Intelligence in Education, Sydney Australia (2003).

[15] Melis, E., E. Andres, A. Franke, A. Frischauf, G. Goguadse, P. Libbrecht, M. Pollet and C. Ullrich, *ActiveMath: A web-based Learning Environment*, IJAIED **12** (2001).

[16] Rowin, C., *Minutes of CETIS Assessment Special Interest Group,* http://assessment.cetis.ac.uk/minutes (2003).

[17] Smythe, C., E. Shepherd, L. Brewer and S. Lay, *IMS question and test interoperability (v 1.2): An overview*, http://www.imsproject.org/.

[18] XForms, http://www.w3.org/MarkUp/Forms/.